Getting Started with PySpark

Srikanth Potukuchi

Visit - https://www.meetup.com/virtual-data-science-workshops/

Prerequisites

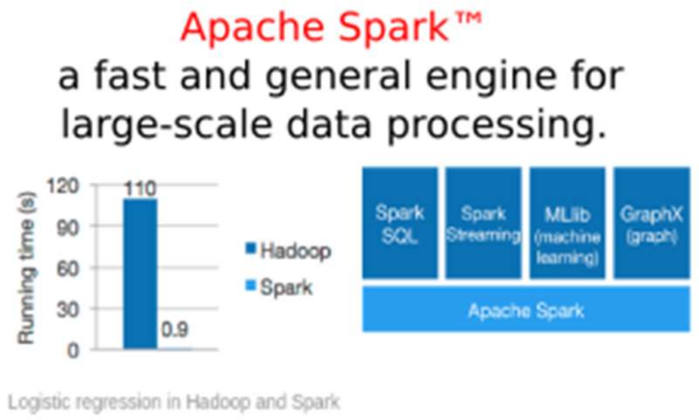
- Signup for a free account with Databricks community edition: https://community.cloud.databricks.com/login.html

Outline

- What is Apache Spark/PySpark?
- PySpark Architecture
- PySpark DataFrame Operations
- Demo in Databricks
- PySpark SQL Functions
- Dask Vs. PySpark

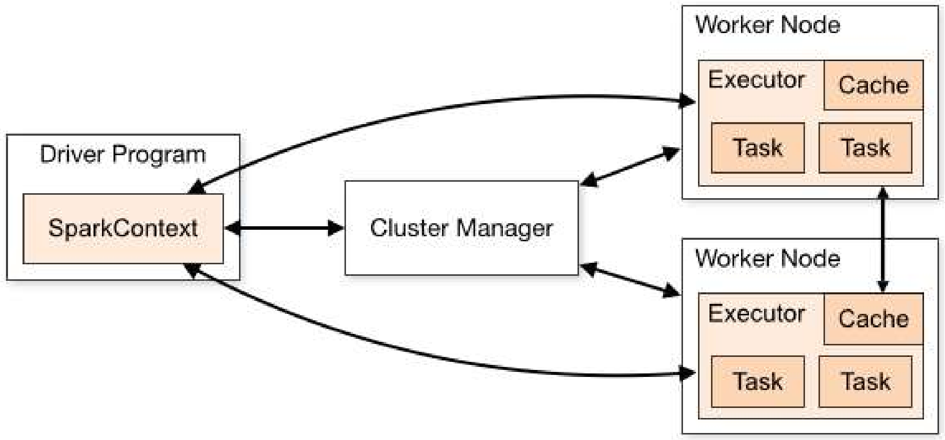
Apache Spark

- Apache Spark™ is a multi-language engine for executing data engineering, data science, and machine learning on single-node machines or clusters. Check - https://spark.apache.org/
- It started as a research project at the UC Berkeley AMPLab in 2009, and was open sourced in early 2010.
- Several languages supported - Python, SQL, Scala, Java, R
- Apache Spark Features:
  - In-memory computation
  - Distributed processing using parallelize
  - Can be used with many cluster managers (Spark, Yarn, Mesos e.t.c)
  - Fault-tolerant
  - Immutable
  - Lazy evaluation
  - Cache & persistence
  - Inbuild-optimization when using DataFrames
  - Supports ANSI SQL



Apache Spark – Architecture

- Apache Spark works in a master-slave architecture where the master is called “Driver” and slaves are called “Workers”. When you run a Spark application, Spark Driver creates a context that is an entry point to your application, and all operations (transformations and actions) are executed on worker nodes, and the resources are managed by Cluster Manager.



PySpark – Dataframe Operations

- PySpark is a Spark library written in Python to run Python applications using Apache Spark capabilities, using PySpark we can run applications parallelly on the distributed cluster (multiple nodes).
- PySpark is a Python API for Apache Spark. Apache Spark is an analytical processing engine for large scale powerful distributed data processing and machine learning applications.

- PySpark DataFrame Creation ways:
    - Manually using toDF() and createDataFrame()
    - data sources like TXT, CSV, JSON, ORV, Avro, Parquet
    - AWS S3 bucket, DBFS, Azure Blob file systems

- Getting Started with PySpark - https://medium.com/geekculture/getting-started-with-pyspark-in-30-minutes-fc9771ab8401

PySpark – SQL Functions

- PySpark provides built-in standard Aggregate functions defines in DataFrame API, these come in handy when we need to make aggregate operations on DataFrame columns. Aggregate functions operate on a group of rows and calculate a single return value for every group.
- Here's a list:
  - approx_count_distinct
  - avg
  - countDistinct
  - count
  - grouping
  - first
  - last
  - kurtosis
  - max
  - min
  - mean
  - skewness
  - stddev
  - sum
  - sumDistinct

Dask Vs. PySpark

- Generally Dask is smaller and lighter weight than Spark. This means that it has fewer features and, instead, is used in conjunction with other libraries, particularly those in the numeric Python ecosystem. It couples with libraries like Pandas or Scikit-Learn to achieve high-level functionality.
- Spark is able to deal with much bigger work loads than Dask. If your data is larger than 1TB, Spark is probably the way to go.
- Dask's SQL engine is premature. Unlike Spark, you can't manipulate your data with SQL queries (for now).
- Spark is part of the Apache eco-system. Unlike Dask, it integrates with other Apache tools such as Hive and Iceberg.
- Spark provides GraphX, a library for graph processing. Dask provides no such library.

- Sources :
  - https://www.slideshare.net/databricks/dask-and-apache-spark
  - https://www.youtube.com/watch?v=hM95j7FNA6M&t=203s&ab_channel=Databricks
  - https://docs.dask.org/en/stable/spark.html
  - https://medium.com/geekculture/dask-or-spark-a-comparison-for-data-scientists-d4cba8ba9ef7
  - https://docs.databricks.com/integrations/graphframes/graph-analysis-tutorial.html